Министерство науки и высшего образования РФ

ФГАОУ ВО Пермский национальный исследовательский  
политехнический университет

Кафедра «Вычислительная математика, механика и биомеханика»

Отчёт по лабораторной работе № 4  
тема «Графический пользовательский интерфейс»  
по дисциплине «Информатика»

Выполнил: студент группы ПМ-23-1б Кузнецов Д.В.

Проверил: ст. пр. каф. ВММБ Ильиных Г.В.

Пермь, 2024

**Задание 1**

* 1. ***Постановка задачи***

Разработать графический пользовательский интерфейс программы определения попадания точки в область. Шаги реализации:

1. График показывает точку и контуры фигуры, заданные в программе. Реализовать цветовую индекацию попадания точки в область

2. Реализовать ввод координат точки через интерфейс

3.Реализовать ввод уравнений кривых через интерфейс

4.Реализовать включение/выключение уравнений

5.Реализовать добавление точки с произвольными коордианатами

* 1. ***Алгоритм выполнения***

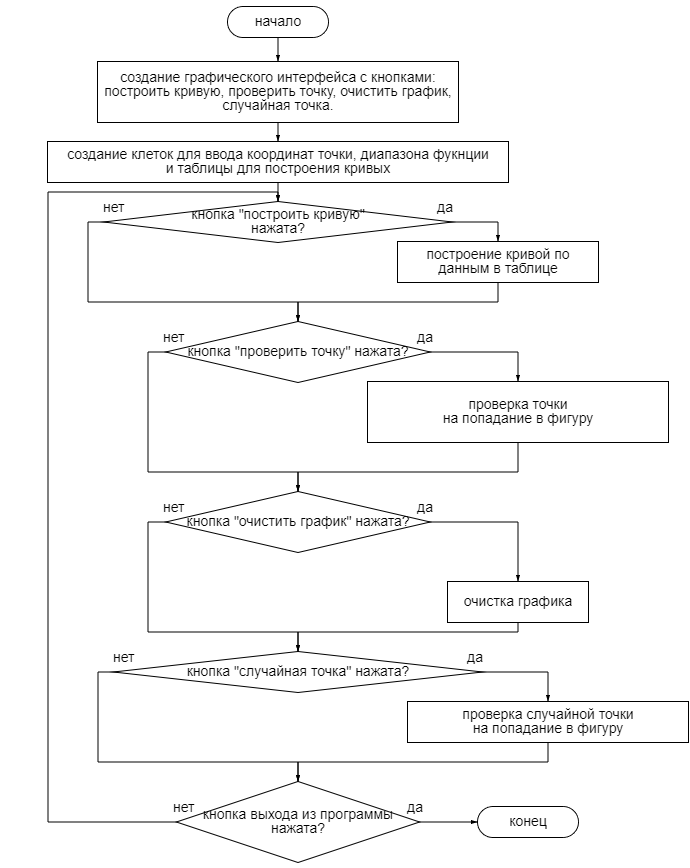


Рис.1 – Блок-схема программы №1

* 1. ***Код программы***

1. import random  
   from PyQt5.QtWidgets import (QApplication, QLabel, QLineEdit, QMainWindow, QPushButton, QFormLayout, QWidget, QTableWidgetItem, QTableWidget, QComboBox)  
   import numpy as np  
   import matplotlib.pyplot as plt  
   from matplotlib.backends.backend\_qt5agg import FigureCanvasQTAgg as FigureCanvas  
     
     
   class MainWindow(QMainWindow):  
    def \_\_init\_\_(self, parent=None):  
    super(MainWindow, self).\_\_init\_\_(parent)  
     
    self.setWindowTitle('График')  
    self.fig = plt.figure()  
    self.canvas = FigureCanvas(self.fig)  
     
    cental\_widget = QWidget()  
    layout = QFormLayout()  
    cental\_widget.setLayout(layout)  
     
    layout.addWidget(self.canvas)  
     
    self.setCentralWidget(cental\_widget)  
     
    self.plot\_button = QPushButton('Проверить точку')  
    self.plot\_button.clicked.connect(self.check2)  
     
    self.scatter\_label = QLabel('Введите координату точки:')  
    self.scatter\_start\_input = QLineEdit('0')  
    self.scatter\_end\_input = QLineEdit('1')  
    self.scatter\_start\_input.setFixedSize(50, 25)  
    self.scatter\_end\_input.setFixedSize(50, 25)  
     
    self.clear\_button = QPushButton('Очистить график')  
    self.clear\_button.clicked.connect(self.clear\_plot)  
     
    self.range\_label = QLabel('Введите диапазон фукнции:')  
    self.range\_start\_input = QLineEdit('0')  
    self.range\_end\_input = QLineEdit('1')  
    self.range\_start\_input.setFixedSize(50, 25)  
    self.range\_end\_input.setFixedSize(50, 25)  
     
    self.higherlower = QComboBox()  
    self.higherlower.addItems(['выше', 'ниже'])  
     
    self.table = QTableWidget()  
    self.table.setColumnCount(4)  
    self.table.setRowCount(1)  
    self.table.setHorizontalHeaderItem(0, QTableWidgetItem('k'))  
    self.table.setHorizontalHeaderItem(1, QTableWidgetItem('степень x'))  
    self.table.setHorizontalHeaderItem(2, QTableWidgetItem('b'))  
    self.table.setHorizontalHeaderItem(3, QTableWidgetItem('выше/ниже'))  
    self.table.setCellWidget(0, 3, self.higherlower)  
    self.start\_button = QPushButton('Нарисовать кривую')  
    self.start\_button.clicked.connect(self.plot\_data1)  
     
    self.random\_button = QPushButton("Случайная точка")  
    self.random\_button.clicked.connect(self.random\_choice)  
    self.vectors\_x = []  
    layout.addRow(self.scatter\_label)  
    layout.addRow(self.scatter\_start\_input, self.scatter\_end\_input)  
    layout.addRow(self.range\_label)  
    layout.addRow(self.range\_start\_input, self.range\_end\_input)  
    layout.addWidget(self.start\_button)  
    layout.addWidget(self.plot\_button)  
    layout.addWidget(self.clear\_button)  
    layout.addWidget(self.random\_button)  
    layout.addRow(self.table)  
     
    def check1(self, x, y, k, n, b, hl):  
    k, n, b = map(float, (k, n, b))  
    if hl == 'выше':  
    return y >= k \* x\*\*n + b  
    else:  
    return y <= k \* x\*\*n + b  
     
    def check2(self):  
    x, y = map(float, [self.scatter\_start\_input.text(), self.scatter\_end\_input.text()])  
    count = 0  
    number = len(self.vectors\_x)  
    for koofs in self.vectors\_x:  
    if self.check1(x, y, koofs[0], koofs[1], koofs[2], koofs[3]):  
    count += 1  
    if count == number:  
    plt.scatter(x, y, color='green')  
    else:  
    plt.scatter(x, y, color='red')  
    self.canvas.draw()  
     
    def vectors\_line(self):  
    k = self.table.item(0, 0).text()  
    n = self.table.item(0, 1).text()  
    b = self.table.item(0, 2).text()  
    expression = f'{k} \* x\*\*({n}) + ({b})'  
    try:  
    range\_start, range\_end = map(float, (self.range\_start\_input.text(), self.range\_end\_input.text()))  
    x = np.linspace(range\_start, range\_end, 50)  
    functions = {}  
    exec(f'def f(x): return {expression}', functions)  
    function = functions['f']  
    y = [function(value) for value in x]  
    return x, y  
    except SyntaxError:  
    return 0  
    except NameError:  
    return 0  
     
    def plot\_data1(self):  
    hl = str(self.table.cellWidget(0, 3).currentText())  
    k = self.table.item(0, 0).text()  
    n = self.table.item(0, 1).text()  
    b = self.table.item(0, 2).text()  
    self.vectors\_x.append([k, n, b, hl])  
    if self.vectors\_line() != 0:  
    x, y = self.vectors\_line()  
    axes = plt.subplot()  
    axes.plot(x, y, color='black')  
    plt.grid(True)  
    plt.xlabel('x')  
    plt.ylabel('y')  
    self.centralWidget().layout().itemAt(0).widget().draw()  
     
    def clear\_plot(self):  
    for ax in self.fig.axes:  
    ax.clear()  
    self.vectors\_x = []  
    self.canvas.draw()  
     
    def random\_choice(self):  
    a = str(random.uniform(-5, 5))[0:4]  
    b = str(random.uniform(-5, 5))[0:4]  
    self.scatter\_start\_input.setText(a)  
    self.scatter\_end\_input.setText(b)  
    self.check2()  
     
   app = QApplication([])  
   main\_window = MainWindow()  
   main\_window.show()  
   app.exec()
   1. ***Тестирование работы программы***

Таблица 1 – Тестирование работы программы №1

|  |
| --- |
| Для проверки была построена фигура из двух кривых на диапазоне [0;1]. Для проверки были введены 3 точки: (1;4), (0,6;3,6), (1;3) и также один раз использовалась случайная точка. |
| Для следующей проверки был создан произвольный треугольник и использовалась функция случайная точка |

**Задание 2**

***2.1 Постановка задачи***

Разработать приложение для визуализации работы метода половинного деления.

Шаги реализации:

1. График показывает кривую в заданном диапазоне и результат работы метода в текстовом виде

2. Реализовать ввод уравнения кривой

3. Реализовать вывод границ и выбранной середины текущего интервала на графике

4. Реализовать анимацию работы метода с помощью дополнительных пауз на каждой итерации

5. Реализовать возможность пошагового нахождения корня

***2.2 Алгоритм решения***

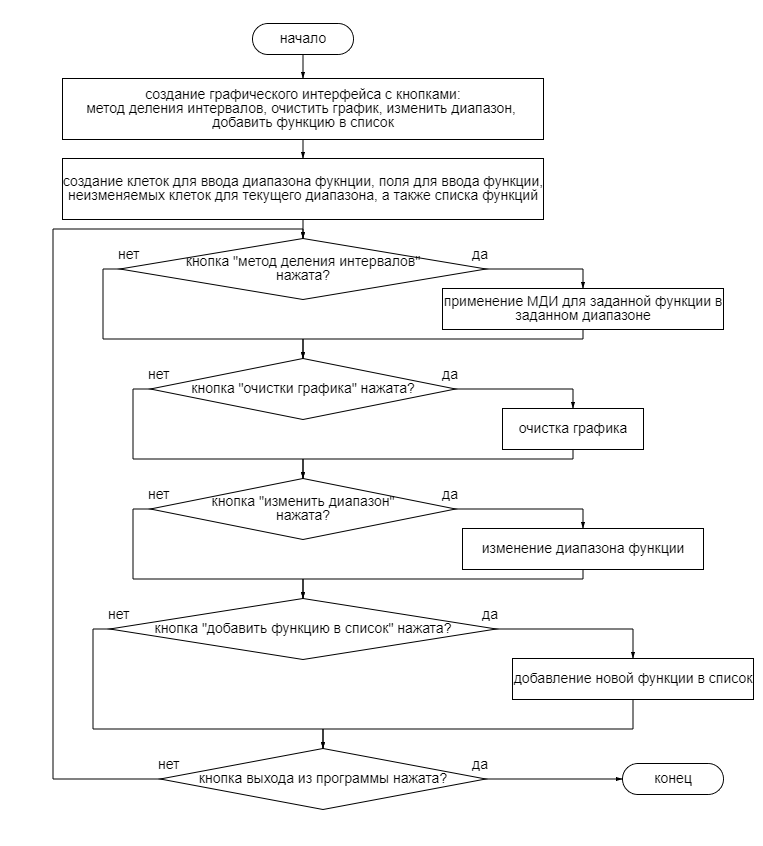


Рис.2 – Блок-схема программы №2

***2.3 Код программы***

from PyQt5.QtWidgets import (QApplication, QLabel, QLineEdit, QMainWindow, QPushButton, QFormLayout, QWidget, QComboBox, QMessageBox)  
import numpy as np  
import matplotlib.pyplot as plt  
from matplotlib.backends.backend\_qt5agg import FigureCanvasQTAgg as FigureCanvas  
from PyQt5 import QtGui  
  
  
class MainWindow(QMainWindow):  
 def \_\_init\_\_(self, parent=None):  
 super(MainWindow, self).\_\_init\_\_(parent)  
  
 self.setWindowTitle('График')  
 self.fig = plt.figure()  
 self.canvas = FigureCanvas(self.fig)  
 self.setWindowIcon(QtGui.QIcon('123.png'))  
  
 cental\_widget = QWidget()  
 layout = QFormLayout()  
 cental\_widget.setLayout(layout)  
 layout.addWidget(self.canvas)  
 plt.grid(True)  
 self.setCentralWidget(cental\_widget)  
  
 self.plot\_button = QPushButton('Нарисовать график')  
 self.plot\_button.clicked.connect(self.plot\_data)  
  
 self.range\_label = QLabel('Диапазон:')  
 self.range\_start\_input = QLineEdit('-10')  
 self.range\_end\_input = QLineEdit('10')  
  
 self.add\_function\_button = QPushButton('Добавить функцию в список')  
 self.function\_input = QLineEdit(' Введите функцию для добавление в список ')  
 self.function\_widget = QComboBox()  
 self.function\_widget.addItems(['x + 3', '2\*x - 5', 'x\*\*3 - 5', ''])  
 self.add\_function\_button.clicked.connect(self.add\_function)  
  
 self.point\_amount = QLabel('Количество точек на графике:')  
 self.point\_input = QLineEdit('50')  
  
 self.clear\_button = QPushButton('Очистить график')  
 self.clear\_button.clicked.connect(self.clear\_plot)  
  
 self.error\_message = QMessageBox()  
 self.error\_message.setText("Функция введена неверно!")  
 self.error\_message.setWindowTitle('Ошибка!')  
 self.error\_message.setIcon(QMessageBox.Critical)  
  
 self.error\_message2 = QMessageBox()  
 self.error\_message2.setText("На данном диапазоне f(c) != 0")  
 self.error\_message2.setWindowTitle('Ошибка!')  
 self.error\_message2.setIcon(QMessageBox.Critical)  
  
 self.bisection\_button = QPushButton('Метод деления интервалов')  
 self.bisection\_button.clicked.connect(self.bisection)  
  
 self.bisection\_label = QLabel("Текущий диапазон:")  
 self.bisection\_range\_start\_current = QLineEdit('-10')  
 self.bisection\_range\_end\_current = QLineEdit('10')  
 self.bisection\_range\_start\_current.setEnabled(False)  
 self.bisection\_range\_end\_current.setEnabled(False)  
  
 self.add\_range\_button = QPushButton('Изменить диапазон')  
 self.add\_range\_button.clicked.connect(self.change\_range)  
 layout.addWidget(self.function\_widget)  
 layout.addWidget(self.range\_label)  
 layout.addWidget(self.range\_start\_input)  
 layout.addWidget(self.range\_end\_input)  
 layout.addWidget(self.add\_range\_button)  
 layout.addWidget(self.bisection\_label)  
 layout.addWidget(self.bisection\_range\_start\_current)  
 layout.addWidget(self.bisection\_range\_end\_current)  
 layout.addWidget(self.bisection\_button)  
 layout.addWidget(self.clear\_button)  
 layout.addRow(self.add\_function\_button, self.function\_input)  
  
 def vectors(self):  
 expression = self.function\_widget.currentText()  
  
 try:  
 range\_start = float(self.range\_start\_input.text())  
 range\_end = float(self.range\_end\_input.text())  
 points = int(self.point\_input.text())  
 except ValueError:  
 range\_start = 0  
 range\_end = 1  
 points = 50  
  
 functions = {}  
 try:  
 exec(f'def f(x): return {expression}', functions)  
 function = functions['f']  
 x = np.linspace(range\_start, range\_end, points)  
 y = [function(value) for value in x]  
 return x, y  
  
 except NameError:  
 self.error\_message.show()  
 return 0  
 except SyntaxError:  
 self.error\_message.show()  
 return 0  
  
 def plot\_data(self):  
 if self.vectors() != 0:  
 x, y = self.vectors()  
 axes = plt.subplot()  
 axes.plot(x, y)  
 plt.grid(True)  
 plt.xlabel('x')  
 plt.ylabel('y')  
  
 self.centralWidget().layout().itemAt(0).widget().draw()  
  
 def clear\_plot(self):  
 for ax in self.fig.axes:  
 ax.clear()  
 plt.grid(True)  
 self.canvas.draw()  
  
 def add\_function(self):  
 text\_x = self.function\_input.text()  
 self.function\_widget.addItems([text\_x])  
  
 def bisection(self):  
 a, b = float(self.bisection\_range\_start\_current.text()), float(self.bisection\_range\_end\_current.text())  
 EPS = 10\*\*-3  
 expression = self.function\_widget.currentText()  
 functions = {}  
 exec(f'def ff(x): return {expression}', functions)  
 f = functions['ff']  
 c = (a + b) / 2  
  
 if f(c) == 0:  
 plt.scatter(c, f(c))  
 self.canvas.draw()  
 self.bisection\_range\_start\_current.setText(str(a))  
 self.bisection\_range\_end\_current.setText(str(b))  
 elif abs(f(c)) < EPS:  
 plt.scatter(c, f(c))  
 self.canvas.draw()  
 if f(c) < 0:  
 if f(b) > 0:  
 a = c  
 elif f(a) > 0:  
 b = c  
 elif f(c) > 0:  
 if f(b) < 0:  
 a = c  
 elif f(a) < 0:  
 b = c  
 x = np.linspace(a, b, 50)  
 y = [f(value) for value in x]  
 plt.plot(x, y)  
 self.bisection\_range\_start\_current.setText(str(a))  
 self.bisection\_range\_end\_current.setText(str(b))  
 self.canvas.draw()  
  
 def change\_range(self):  
 self.bisection\_range\_start\_current.setText(str(self.range\_start\_input.text()))  
 self.bisection\_range\_end\_current.setText(str(self.range\_end\_input.text()))  
  
  
app = QApplication([])  
main\_window = MainWindow()  
main\_window.show()  
app.exec()

***2.4 Тестирование работы программы с проверкой***

Для проверки построения графика используется сайт Wolfram Alpha.

Таблица 2 - Тестирование работы и проверка решений программы №2

|  |
| --- |
|  |
|  |
|  |

**Задание 3  
*3.1 Постановка задачи***

Разработать приложение для построения примитивной картографии

Шаги реализации:

1. Отобразить заданные в программе объекты разными маркерами, добавить легенду для расшифровки.

2. Создать таблицу для внесения координат и типов объектов инфраструктуры (насосные станции, пожарные водоемы, и т.п.), для выбора типа объекта предусмотреть выпадающее меню.

3. Отобразить на полотне участки. Создать таблицу для внесения данных по координатам участков (участки задаются как прямоугольники, координатами н.л. угла, шириной, длиной и углом поворота)

4. Добавить для водоемов радиус и отобразить их в виде кругов

***3.2 Алгоритм решения***

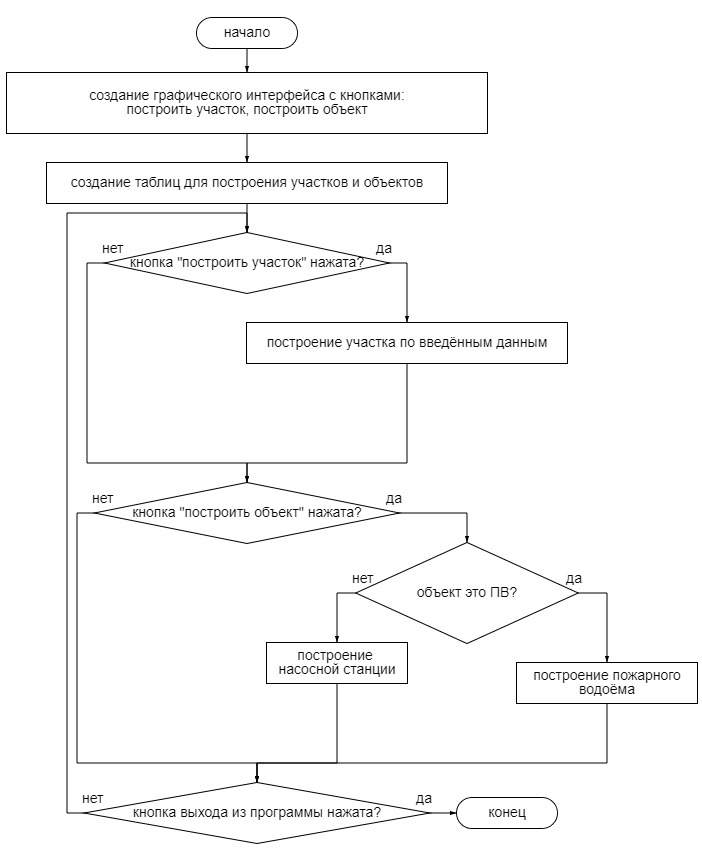


Рис.2 – Блок-схема программы №3

***3.3 Код программы***

from PyQt5.QtWidgets import (QApplication, QLabel, QLineEdit, QMainWindow, QPushButton, QFormLayout, QWidget, QComboBox, QMessageBox, QTableWidget, QTableWidgetItem)  
import matplotlib.pyplot as plt  
import matplotlib.patches as ptch  
from matplotlib.backends.backend\_qt5agg import FigureCanvasQTAgg as FigureCanvas  
from PIL import Image  
im = Image.open('whitewallpaper.png')  
class MainWindow(QMainWindow):  
 def \_\_init\_\_(self, parent=None):  
 super(MainWindow, self).\_\_init\_\_(parent)  
  
 self.setWindowTitle('Задание 3')  
 self.fig = plt.figure()  
 self.canvas = FigureCanvas(self.fig)  
  
 cental\_widget = QWidget()  
 layout = QFormLayout()  
 cental\_widget.setLayout(layout)  
  
 self.setCentralWidget(cental\_widget)  
  
 self.life\_table = QTableWidget()  
 self.life\_table.setColumnCount(5)  
 self.life\_table.setRowCount(1)  
 self.life\_table.setHorizontalHeaderLabels(['x', 'y', 'l', 'h', 'a'])  
 self.life\_table.setFixedWidth(525)  
 self.life\_table.setFixedHeight(50)  
  
 self.object\_table = QTableWidget()  
 self.object\_table.setColumnCount(4)  
 self.object\_table.setRowCount(1)  
 self.object\_table.setHorizontalHeaderLabels(['Тип объекта', 'x', 'y', 'r'])  
 self.object\_table.setFixedWidth(450)  
 self.object\_table.setFixedHeight(50)  
 self.start\_button = QPushButton('Построить участок')  
 self.start\_button.clicked.connect(self.plotd)  
 self.start\_button.setFixedWidth(450)  
  
 self.object\_button = QPushButton('Построить объект')  
 self.object\_button.clicked.connect(self.object\_plot)  
 self.object\_button.setFixedWidth(450)  
  
 self.axes = plt.subplot()  
 self.axes.imshow(im)  
 plt.grid(True)  
  
 layout.addRow(self.life\_table, self.object\_table)  
 layout.addRow(self.canvas)  
 layout.addRow(self.start\_button, self.object\_button)  
  
 for i in range(self.object\_table.rowCount() + 1):  
 self.type\_box = QComboBox()  
 self.type\_box.addItems(['ПВ', 'НС'])  
 self.object\_table.setCellWidget(i, 0, self.type\_box)  
  
 def plotd(self):  
 x, y = map(int, (self.life\_table.item(0, 0).text(), self.life\_table.item(0, 1).text()))  
 l, h = map(int, (self.life\_table.item(0, 2).text(), self.life\_table.item(0, 3).text()))  
 ang = int(self.life\_table.item(0, 4).text())  
 rect = ptch.Rectangle((x, y), l, h, linewidth=1, edgecolor='black', facecolor='none', angle=ang)  
 self.axes.add\_patch(rect)  
 self.canvas.draw()  
  
 def object\_plot(self):  
 obj = str(self.object\_table.cellWidget(0, 0).currentText())  
 if obj == 'НС':  
 self.build\_circle()  
 else:  
 self.build\_scatter()  
  
 self.canvas.draw()  
  
 def build\_circle(self):  
 x, y = map(int, (self.object\_table.item(0, 1).text(), self.object\_table.item(0, 2).text()))  
 r = int(self.object\_table.item(0, 3).text())  
 circle = ptch.Circle((x, y), radius=r, color='green')  
 self.axes.add\_patch(circle)  
  
 def build\_scatter(self):  
 x, y = map(int, (self.object\_table.item(0, 1).text(), self.object\_table.item(0, 2).text()))  
 self.axes.scatter(x, y, color='black')  
  
  
app = QApplication([])  
main\_window = MainWindow()  
main\_window.show()  
app.exec()

***3.4 Тестирование работы программы с проверкой***

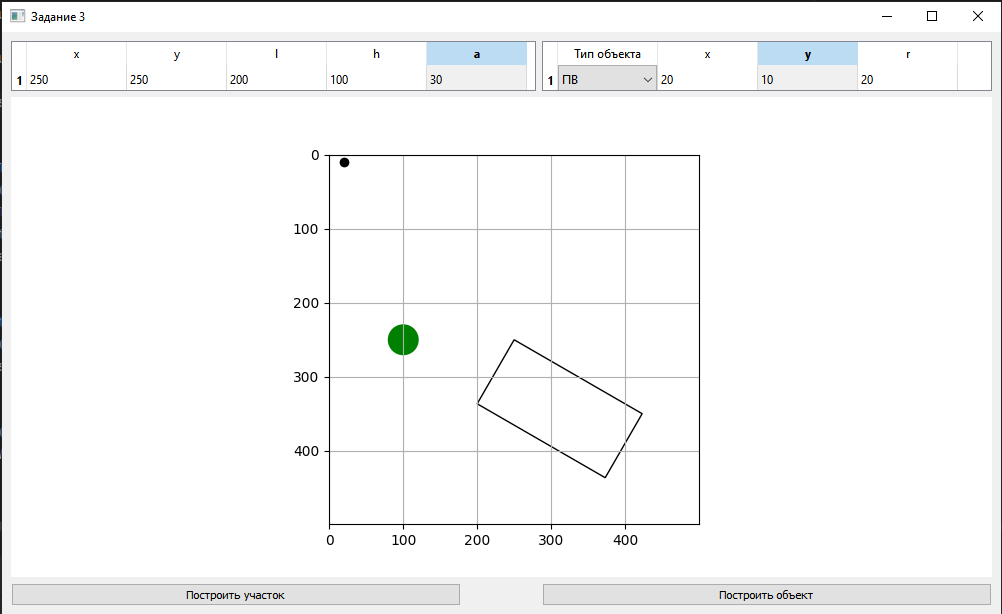


Рис.4 – Тестирование работы программы №3